(12) UK Patent Application (19) GB (11) 2 337 834 (13) A

(43) Date of A Publication 01.12.1999

(21) Application No 9804910.9

(22) Date of Filing 06.03.1998

(51) INT CL[6]
G06F 11/00 11/34

(52) UK CL (Edition Q)
G4A AFMP AFMT

(56) Documents Cited
EP 0854422 A2 EP 0849672 A2

(58) Field of Search
UK CL (Edition Q) G4A AFMP AFMT
INT CL[6] G06F 11/00 11/34

(71) Applicant(s)
LSI Logic Corporation
(Incorporated in USA - Delaware)
1551 McCarthy Blvd, Milpitas, California 95035,
United States of America

(72) Inventor(s)
Simon Kershaw
Graham Kirsch

(74) Agent and/or Address for Service
Miller Sturt Kenyon
9 John Street, LONDON, WC1N 2ES, United Kingdom

(54) Abstract Title
Processor debugging using scan in-circuit emulation

(57) Entry into a debugging operation using a scan chain of registers in a microprocessor, includes the following steps:-

1. Providing the processor 40 with a chain of scan registers 16, scan interface logic for interfacing with an external scan controller, a breakpoint interrupt means for executing an interrupt instruction, and a processor clock control unit 46;

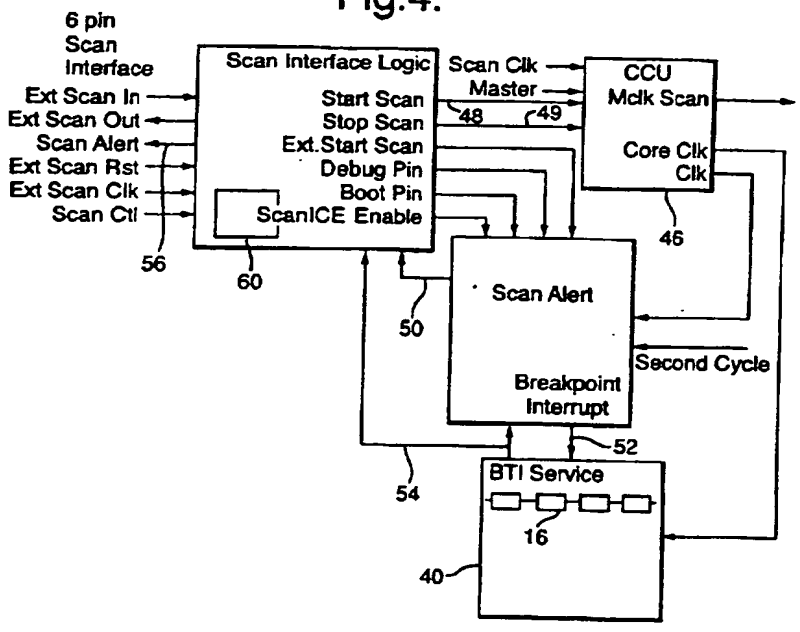
2. Detecting or generating a breakpoint in the operation of the processor;

3. The breakpoint interrupt means executes an interrupt instruction as a result of which the processor completes its current instruction, and signals the same to the scan interface logic;

4. The scan interface logic asserts a Start Scan signal to the clock control unit, which whereupon stops the processor clock or clocks; and

5. The external scan controller is alerted to start a scan sequence.

Fig.4.



GB 2 337 834 A

Fig.1.

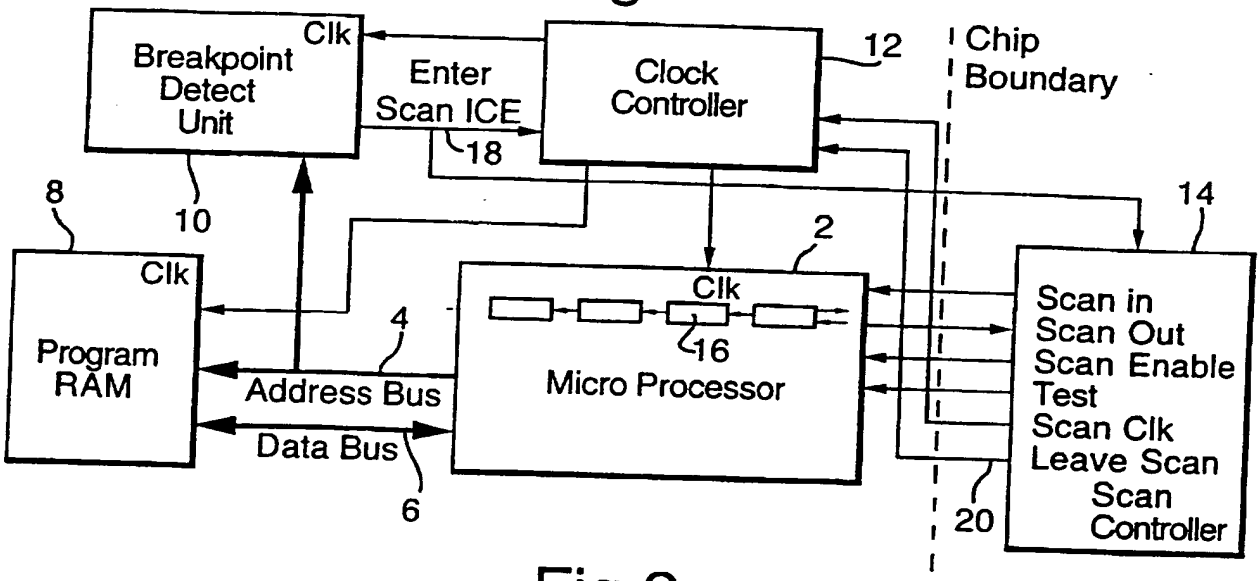


Fig.2.

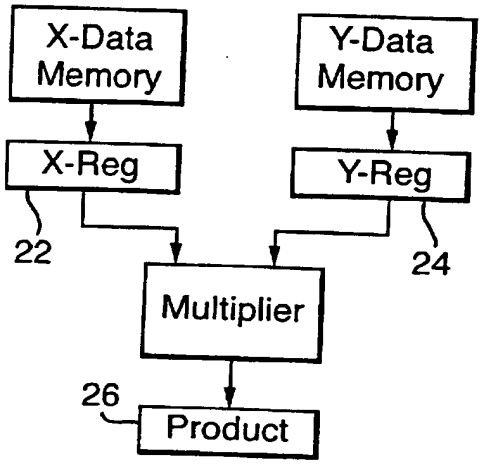


Fig.3.

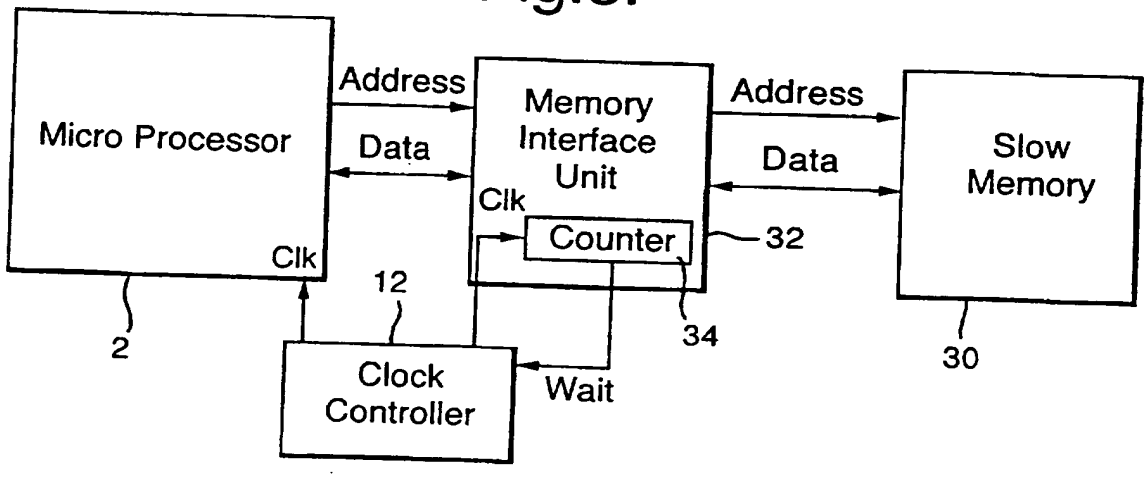


Fig.4.

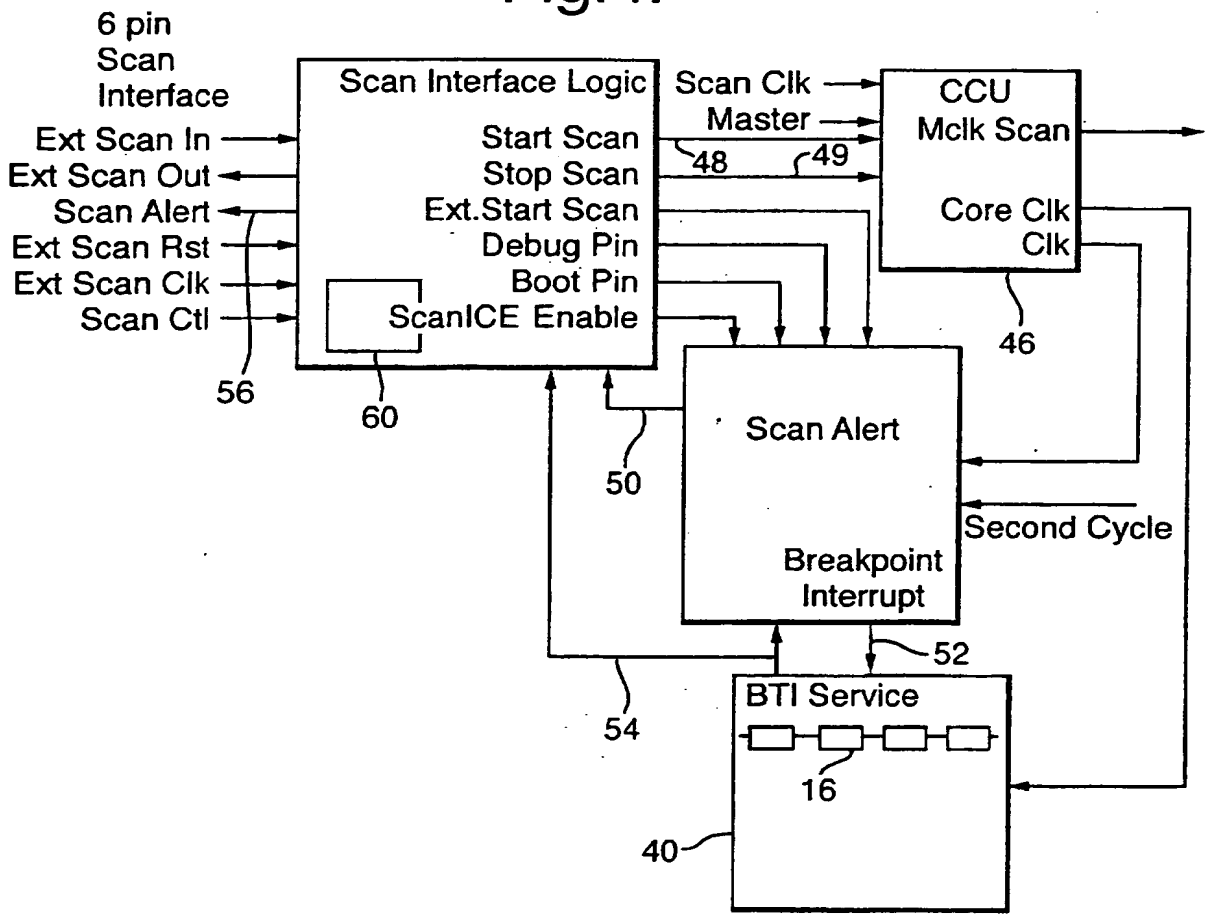

6 pin
Scan
Interface
Ext Scan In
Ext Scan Out
Scan Alert
Ext Scan Rst
Ext Scan Clk
Scan Ctl
Scan Interface Logic
Start Scan
Stop Scan
Ext.Start Scan
Debug Pin
Boot Pin
ScanICE Enable
Scan Clk
Master
48
49
CCU
Mclk Scan
Core Clk
Clk
46
56
60
50
Scan Alert
Second Cycle
Breakpoint
Interrupt
52
54
BTI Service
16
40

Fig.7.

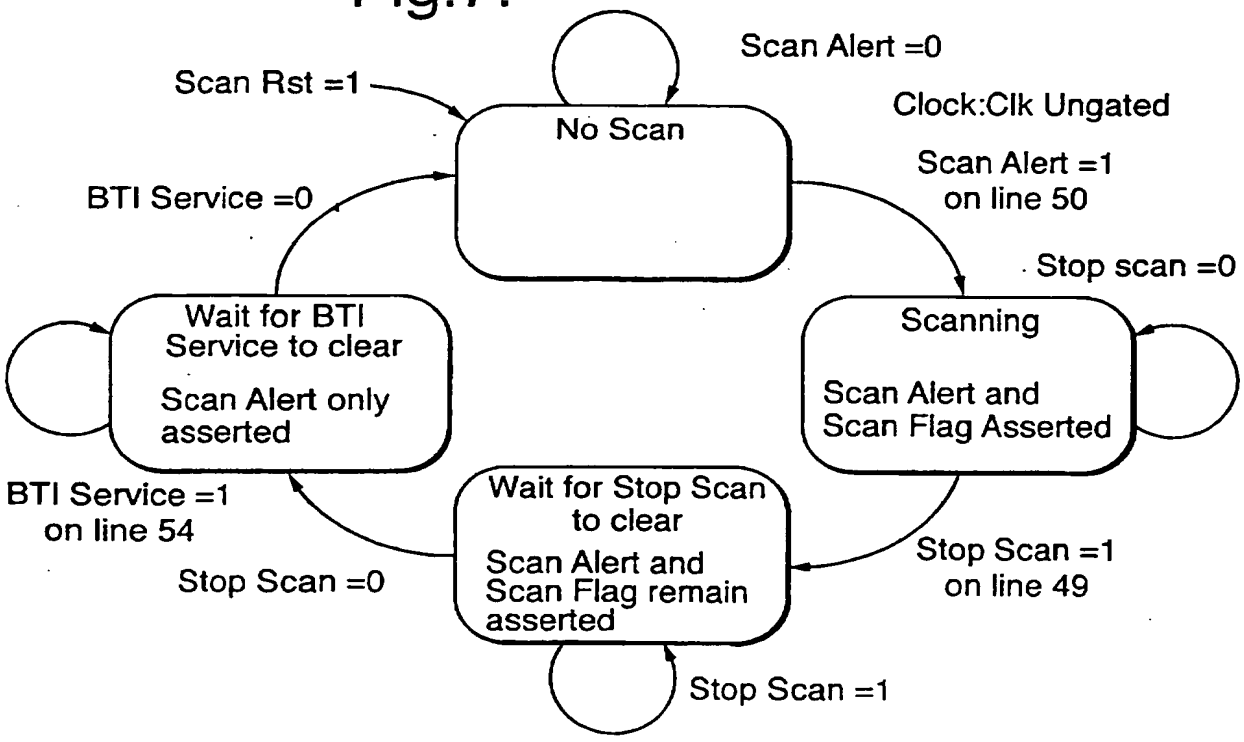

Scan Alert =0
Scan Rst =1
No Scan
Clock:Clk Ungated
Scan Alert =1
on line 50
BTI Service =0
Stop scan =0
Wait for BTI
Service to clear
Scan Alert only
asserted
Scanning
Scan Alert and
Scan Flag Asserted
BTI Service =1
on line 54
Wait for Stop Scan
to clear
Scan Alert and
Scan Flag remain
asserted
Stop Scan =1
on line 49
Stop Scan =0
Stop Scan =1

Fig.5.

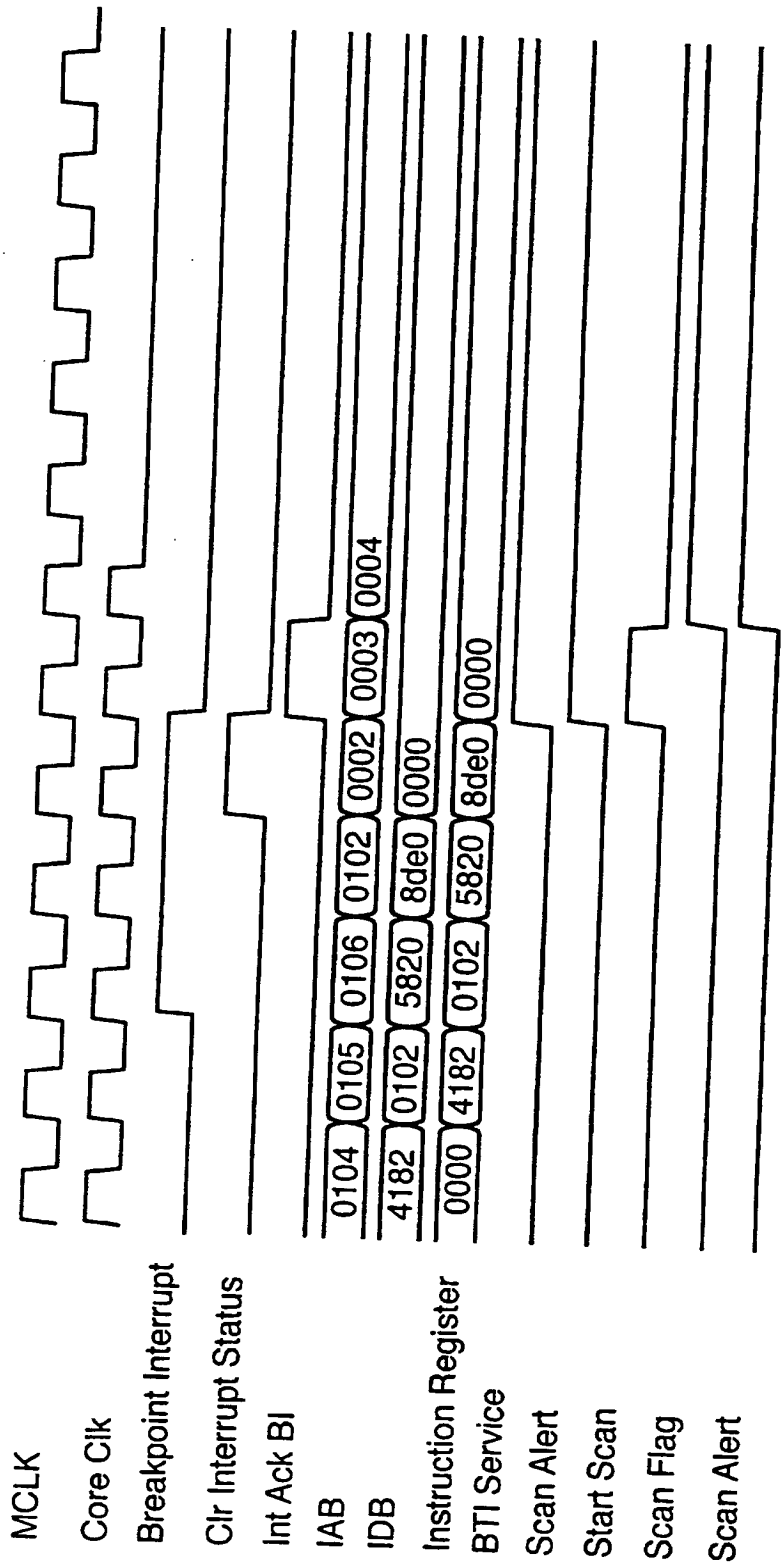
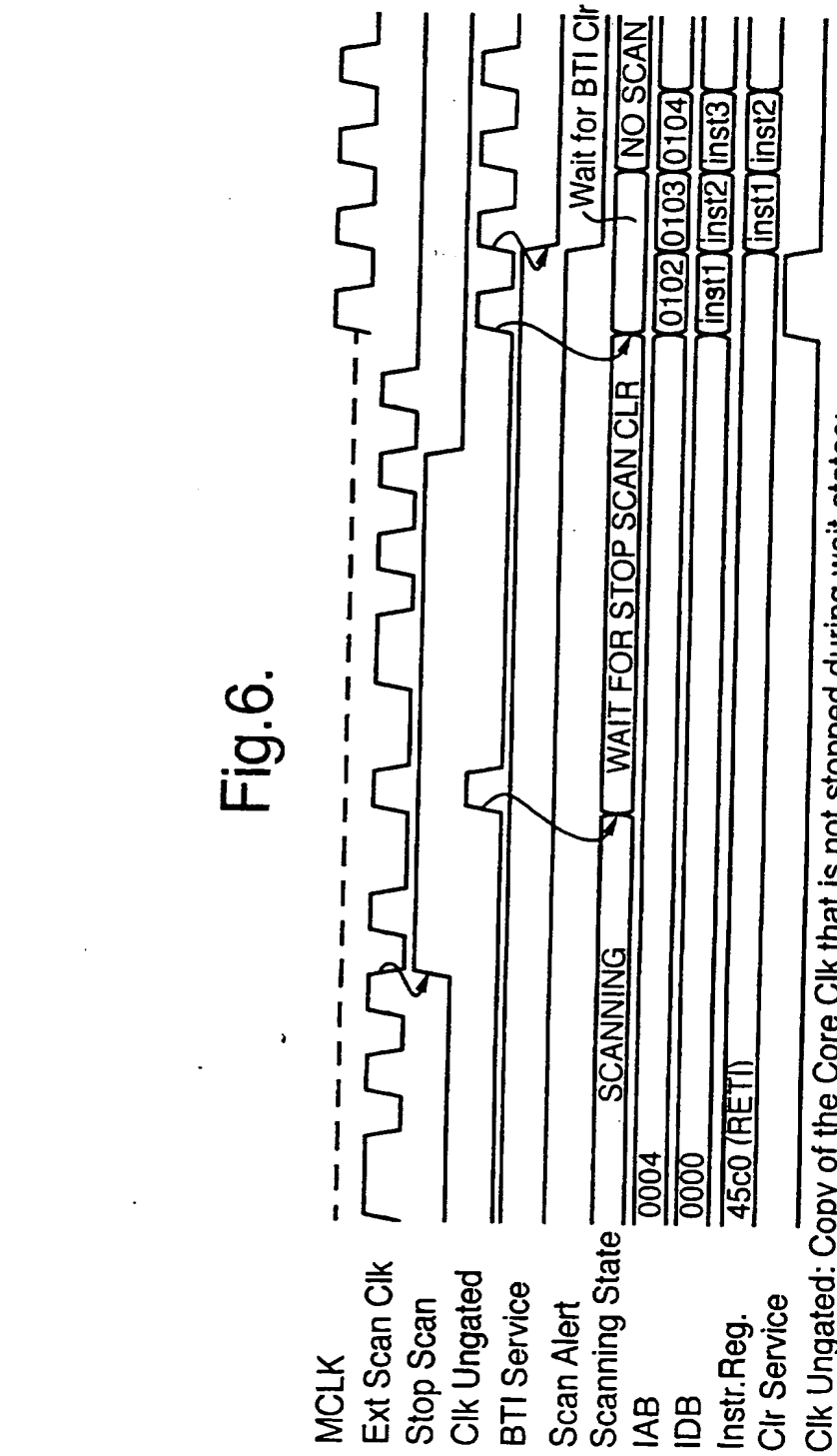

MCLK
Core Clk
Breakpoint Interrupt
Clr Interrupt Status
Int Ack BI
IAB
IDB
Instruction Register
BTI Service
Scan Alert
Start Scan
Scan Flag
Scan Alert
0104
0105
0106
0102
0002
0003
0004
4182
0102
5820
8de0
0000
0000
4182
0102
5820
8de0
0000

Fig.6.


MCLK
Ext Scan Clk
Stop Scan
Clk Ungated
BTI Service
Scan Alert
Scanning State
IAB
IDB
Instr.Reg.
Clr Service
Clk Ungated: Copy of the Core Clk that is not stopped during wait-states;
SCANNING
WAIT FOR STOP SCAN CLR
Wait for BTI Clr
NO SCAN
0004
0000
45c0 (RETI)
0102
0103
0104
inst1
inst2
inst3
inst1
inst2

MICROPROCESSOR DEBUGGING

The present invention relates to a method and apparatus for carrying out debugging procedures on a processor, for example a microprocessor or Digital Signal Processing (DSP) processor.

Software debugging is commonly carried out using In Circuit Emulation (ICE) wherein a monitor program located in the microprocessor provides information to an external host.

It has been recognised that software debugging using ICE techniques may be carried out more expeditiously using techniques adapted from production testing of microprocessors with production scan-chains. Such production scan chains are to be distinguished from boundary scan registers as in the known JTAG standards. In production scan chains, registers are provided throughout the processor so that the working of a software routine throughout the processor can be observed, i.e. it is "visible". Such production scan chain of registers can be loaded with a random pattern of logic values. One or more machine cycles may then be executed, and the logic values are fed into the microprocessor logic. The resultant logic values in the registers may then be uploaded and examined to assess whether the microprocessor logic is working correctly.

Referring now to figure 1 of the drawings, this shows a block diagram of a previously proposed technique for using an existing production test scan-chain in a microprocessor for software debugging. As shown in figure 1, a pipelined microprocessor 2 is coupled via an Address Bus 4 and Data Bus 6 to a program RAM 8. The Address Bus 4 is also coupled to a Breakpoint Detect Unit 10. A clock controller 12 is provided and a Scan Controller 14 external to the microprocessor controls the operation of the scan mode.

A chain of internal scan registers 16 is such that the state of the microprocessor is completely described in the registers. Further the registers may be accessed during normal operation of the microprocessor.

The method of software debug using a production scan-chain is referred to herein as Scan In-Circuit Emulation (ScanICE). ScanICE mode is entered whenever access is required to the internal processor state, such as on reaching a software breakpoint. Figure 1 illustrates a previous approach to entering ScanICE mode where only breakpoints on program addresses are considered. A register in the Breakpoint Detect Unit 10 is initialised to the required breakpoint address. When a match occurs between the Program Address Bus 4 and this register, an Enter ScanICE signal on a line 18 is asserted. This triggers the Clock Controller 12 to halt appropriate clocks so that the off-chip Scan Controller 14 can initiate scanning. The Scan Controller 14 uses the Scan Enable and Test signals to configure registers in the scan-chain 16 to shift serially when clocked and supplies the Scan Clock.

Using the scanned-out data for observability and scanned-in data for controllability the controller 14 can provide all required facilities for debugging software running on the microprocessor. When ScanICE functions are complete, Leave Scan is asserted on a line 20 forcing the Clock Controller to restart clocks and resume normal operation. The approach of figure 1 for entering ScanICE mode suffers from a number of problems:

1. To ease software development it is often required that the microprocessor's pipeline is hidden from the programmer. This means that all registers' contents refer to the same instruction. In a processor with, for example, a data pipeline, simply stopping the clock will result in several instructions being in varying states of execution. This is illustrated in figure 2 for a simple multiplier data path. The pipeline is used to separate the operation of reading from memory and actually multiplying so that they can be carried out in parallel using the X and Y registers 22,24 as a temporary store (which are part of the production scan chain). For successive single-cycle multiply instructions, two instructions are being executed at any one time; the result of one is loaded into the

product register 26 on a particular clock-edge and the data operands of the next are loaded into the X and Y registers 22,24. By simply stopping the clock the Host debugger will have great difficulty in separating the instructions with just the knowledge of what is held in the X, Y and Product registers.

2. It is often required that a processor be able to access slow-memory without using a very slow clock. One way of achieving this is to provide a wait-state mechanism whereby the processor is halted during the memory access. This is illustrated in figure 3.

Following an access to a slow memory 30, a Memory Interface Unit 32 asserts a Wait signal and Clock Controller 12 stops the clock to the Microprocessor 2. Following a defined number of clocks of the Memory Interface Unit 32 (as monitored by a counter 34) the Wait signal is released and the clock to the Microprocessor is started again and normal operation resumes. When a breakpoint is detected, the Clock Controller must detect the fact that an access to slow memory is occurring and continue clocking the Memory Interface Unit to complete the access before stopping all clocks. This is an undesirable complication.

Depending on the functionality of the microprocessor, there are many other scenarios when just stopping the main clock and entering ScanICE may cause problems. These problems get more numerous as pipelines get more complex and more elaborate schemes are implemented for improving microprocessor performance.

Summary of the Invention

The present invention provides apparatus for carrying out debugging procedures on a processor, the processor including a production scan chain of scan registers, scan interface means for interfacing with a scan controller means external of the processor, a breakpoint interrupt means for executing an interrupt instruction and a processor clock control means, wherein, when in the operation of the processor a breakpoint is detected or generated, the breakpoint interrupt means executes an interrupt, the processor completes its current instruction and then branches to a Interrupt Service Vector, completion of which is signalled to the scan interface means which provides a Start Scan

signal to the clock control means, following which the scan interface means signals the external scan controller to begin a scanning operation.

In a further aspect, the present invention provides a method for carrying out debugging procedures on a processor, the method comprising the following steps:

1) providing a processor with a production scan chain of scan registers, a scan interface means for interfacing with an external scan controller means, a breakpoint interrupt means for executing an interrupt instruction, and a processor clock control means;

2) detecting or generating a breakpoint in the operation of the processor;

3) the breakpoint interrupt means executes an interrupt instruction as a result of which the processor completes its current instruction, and signals the same to the scan interface means;

4) the scan interface means asserts a Start Scan signal to the processor clock control means, which whereupon stops the processor clock or clocks; and

5) the external scan controller means is alerted to start a scan sequence.

In accordance with the invention, an improved technique is provided for entering an In Circuit Emulation session which avoids problems experienced in stopping the microprocessor clock without upsetting current actions of the microprocessor, and in recovering information of pipelined instructions.

Preferably, in the operation of the invention, after the microprocessor has completed its current instructions, the microprocessor pipeline is filled with no operation (NOP) instructions.

The present invention is particularly applicable to pipelined processors, especially DSP processor cores.

Brief Description of the Drawings

A preferred embodiment of the invention will now be described with reference to the accompanying drawings, wherein:-

Figure 1 is a block diagram of a prior approach for entering In Circuit Emulation mode;

Figure 2 is a flow chart of a known method of operation of pipelined instructions;

Figure 3 is an example of a known technique of accessing slow operation memory;

Figure 4 is a block diagram of a preferred embodiment of the invention;

Figures 5 and 6 are timing diagrams of the operation of the embodiment of figure 4 for entering and leaving an ICE mode; and

Figure 7 is a state diagram of a Finite State Machine in the Scan Interface Logic of Figure 4.

Description of the Preferred Embodiment

A preferred embodiment will now be described with reference to figure 4. In figure 4, a core 40 of a processor is shown, comprising a pipelined DSP processor and including a production scan chain of scan registers 16. In this embodiment, the scan registers are formed from existing registers in the processor by coupling a respective multiplexer to the input of the register to permit either the normal operation of the register or to use it in the scan mode wherein test logic values are loaded into the register. All processors, have a mechanism for "interrupting" the flow of instructions. The instigation of the interrupt can take numerous forms including processor input signals and software instructions. Typically, when an interrupt is received, the current instruction being executed is completed and the processor branches to an Interrupt Service Vector (ISV), a dedicated area of memory that determines the appropriate course of action. It is usual for a branch instruction to be located at the ISV to an Interrupt Service Routine (ISR) that contains the interrupt handler. At the end of the ISR is some form of RETurn-from-Interrupt (RETI) instruction that indicates the end of the interrupt service routine and causes the program to resume operation from the point where the interrupt occurred. Processor 40 has one particular interrupt instruction suitable for breakpoint implementation, Breakpoint Interrupt BI/TRAP, that can be instigated via an external input (BI) or a software instruction (TRAP). The following actions occur on the Core when a BI/TRAP is received.

Stack Pointer→Stack Pointer -1

Program Counter→Stack

<ISV Address=0x0002>→Program Counter (forces branch to ISV)

Interrupt Acknowledge signal asserted

It is seen that the program counter is stored on the stack, for use at the end of the ISR.

The processor has a dedicated module 42 that performs a Breakpoint Detect function. This monitors the activity at core 40 and determines when program execution should be halted and the state of the core communicated to an off-chip debugger. The module 42 uses the BI/TRAP interrupt to stop the operation of the core 40.

The action of interrupting the processor using a Breakpoint with an appropriate ISV achieves two important goals:

1. The clock can be stopped during the execution of a known instruction (the Breakpoint ISV);
2. Following the ScanICE service, the state of the processor can be restored by executing a RETurn from Interrupt.

Good values for the ISV are clearly No-Operations (NOP). If the ISV is mapped to zero wait-stated program memory, these NOPs can be simply loaded into the appropriate memory locations otherwise they can be forced into the instruction pipeline using dedicated hardware inside or outside the core. By stopping the clock following the execution of a sufficient numbers of NOPs, the program pipeline is cleared and the state of the data pipeline is guaranteed to represent the last program instruction.

In figure 4, a scan interface logic 44 and a Clock Control Unit (CCU) 46 are provided. Logic 44 is coupled to the CCU 46 by various control lines, including a Start Scan signal line 48, and a Stop Scan signal on line 49. Scan Interface Logic 44 also provides control signals to an external scan controller (not shown), including a Scan Alert signal on line 56. The operation of Scan Interface 44 is determined by a Finite State Machine 60, the state diagram of which is shown in Figure 7.

In operation, the sequence of events is as follows:-

1. When the module 42 either detects a breakpoint or is instructed to stop program execution by the Scan Interface 44, it asserts a Breakpoint Interrupt input on line 52 to the core 40.

2. The core completes its current instruction and then branches to the BI/TRAP ISV where NOPs are either read or forced into the instruction pipeline. BTI Service is asserted at this time by the Core on line 54 to module 42 and Scan Interface 44.

3. When the module 42 detects BTI Service being asserted (and is configured in ScanICE debug mode) it asserts module Scan Alert on line 50 to Scan Interface 44.

4. The Scan Interface 44 is triggered by the assertion of module Scan Alert to begin entry to ScanICE mode. A key part of this is the assertion of Start Scan on line 48.

5. The CCU 46 detects the assertion of Start Scan and cleanly stops the appropriate clocks (including the clock to the Core).

6. Finally, the Scan Interface 44 asserts one of the 6 Scan Interface signals, Scan Alert, on line 56 to indicate to the off-chip Scan Controller that the clocks have been stopped and it should take control. The Master clock is replaced by Ext Scan Clk (another of the signals in the 6 pin Scan Interface) as the source for the clock signals.

A timing diagram illustrating this protocol is included in figure 5. Scan Flag is used internally to indicate ScanICE servicing in progress.

In figure 5, Clr Interrupt Service refers to the core decoder detecting an interrupt request and having completed decoding the previous instruction, it drives the appropriate control signals to service an interrupt. To indicate this special decode function and thus signal the start of the interrupt service routine, Clr Interrupt Service is asserted for one cycle.

IAB : Instruction address bus - to/from program RAM;

IDB : Instruction data bus - to/from program RAM;

It may be seen from Figure 5 that 3 machine cycles following assertion of Breakpoint Interrupt on line 52, BTI Service on line 54 and module scan alert on line 50 are asserted. The data on IDB bus is zero after 2 machine cycles, with a NOP

instruction (0004) on the IAB after 4 cycles. The Scan Alert signal on line 56 is asserted one cycle after BTI Service.

Exit from ScanICE is instigated by serially-loading a RETI instruction into the microprocessor's instruction register during the final full-scan and an appropriate instruction into the Scan Interface Logic's control register. The Scan Interface Logic instigates restarting of the microprocessor's clock, thus executing the RETI instruction and returning to normal program operation. It is clearly important that during the final full-scan when the RETI instruction is loaded, the state of the core is updated correctly. This may simply involve reloading all registers with the values they held prior to entering ScanICE, or with new values as determined by the debugger.

One of the bits in the Scan Interface Logic instruction register is asserted to force exit from ScanICE mode. This "Ctl Stop Scan" bit causes the Stop Scan signal from the Scan Interface to be asserted, and triggers the CCU to restart the stopped clocks. The first instruction that must be executed is a RETurn-from-Interrupt that restores the state of the processor. This is forced into the instruction pipeline by asserting the appropriate bits (45c0) during the final full-scan. The instruction register is effectively loaded directly via the scan-chain. A second timing diagram illustrating the protocol for leaving ScanICE mode is depicted in figure 6. An important feature of the protocol for leaving ScanICE is the requirement that Stop Scan on line 49 is cleared before the clocks are restarted. This ensures clean operation if a breakpoint is required immediately after leaving scan (e.g. single-stepping). The extra Ext Scan Clk required during the assertion of Stop Scan is needed in the CCU to register the changing value of the Stop Scan. Following clearance of Stop Scan on line 49, BTI service on line 54 is cleared when the first instruction Inst1 is present on the core 40.

In the implementation of ScanICE, the protocols for entering and leaving scan are controlled by a Finite State Machine (FSM) 60 in the Scan Interface Logic 44. The state-diagram is illustrated in figure 7 and clarifies the operation discussed above.

A principal advantage of this method is to enable clean entry and exit to and from ScanICE mode. The alternative of simply stopping the clock leads to highly complex

problems to correctly recover the programming model from the microprocessor's registers. The method also prevents the potentially destructive effects of halting access to off-core memories and peripherals.

CLAIMS

1. Apparatus for carrying out debugging procedures on a processor, the processor including a chain of scan registers, scan interface means for interfacing with a scan controller means external of the processor, a breakpoint interrupt means for executing an interrupt instruction, and a processor clock control means wherein, when, in the operation of the microprocessor a breakpoint is detected or generated, the breakpoint interrupt means executes an interrupt, the microprocessor completes its current instruction and then branches to an Interrupt Service Vector, the completion of which is signalled to the scan interface means which provides a Start Scan signal to the clock control means, following which the scan interface means signals the external scan controller to begin a scanning operation.

2. Apparatus according to claim 1, wherein the processor includes a pipeline, and the breakpoint interrupt means includes means for filling the pipeline with no operation (NOP) instructions.

3. Apparatus according to claim 1 or 2, wherein the processor comprises a DSP processor core.

4. Apparatus according to claim 1 or 2, wherein said scan interface means includes a finite state machine for controlling the logical state of the scan interface means.

5. Apparatus according to any preceding claim, including a signal line connecting the processor to the breakpoint interrupt means and scan interface means for asserting a breakpoint condition.

6. Apparatus according to any preceding claim, including a signal line connecting said scan interface means to said clock control means for asserting a Stop Scan signal.

7. Apparatus according to claims 5 and 6, wherein when the processor exits from the scanning mode, the stop scan signal is asserted while the processor returns to normal operation, at which time the breakpoint condition signal is cleared.

8. Apparatus according to any preceding claim, wherein one or more scan registers have a respective multiplexer coupled to the input whereby to permit use of the register in the normal operation of the processor or use in a debugging procedure.

9. A method for carrying out debugging procedures on a processor, the method comprising the following steps:

1. providing a processor with a chain of scan registers, a scan interface means for interfacing with an external scan controller means, a breakpoint interrupt means for executing an interrupt instruction, and a processor clock control means;

2. detecting or generating a breakpoint in the operation of the processor;

3. the breakpoint interrupt means executes an interrupt instruction as a result of which the processor completes its current instruction, and signals the same to the scan interface means;

4. the scan interface means asserts a Start Scan signal to the clock control means, which whereupon stops the processor clock or clocks; and

5. the external scan controller means is alerted to start a scan sequence.

10. A method according to claim 9, wherein in step 3) following execution of the current instruction, a processor pipeline is filled with no operation (NOP) instructions.

11. A method according to claim 9, wherein when the scan sequence is to be terminated, a stop scan signal is asserted to the clock control means by the scan interface means, and a return from interrupt instruction is executed, the processor signalling return to normal operation to said scan interface means.

The Patent Office

12

INVESTOR IN PEOPLE

Application No: GB 9804910.9
Claims searched: 1

Examiner: Leslie Middleton
Date of search: 23 September 1999

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): G4A (AFMT, AFMP)

Int Cl (Ed.6): G06F 11/00 11/34

Other: ONLINE: EPODOC, PAJ, WPI / EPOQUE

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
E,X	EP 0854422 A2	(Texas Ins.) See ll.47-55 p.3, espcly.	1 at least
E,X	EP 0849672 A2	(Texas Ins.) See Claim 1, espcly.	1 at least

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.